Cassandra分享一

基本概念

吴小宁

分享内容

- 基本概念
- 案例分享
- 数据模型
- Repair
- 集群监控
- 性能调优

哪些公司在用？

- Apple，10PB，75,000节点
- Netflix，420TB，2,500节点，每天1万亿请求
- 中国宜搜，300TB，270节点，每天8亿请求
- eBay，250TB，超过100节点

SCALABLE

Some of the largest production deployments include Apple's, with over 75,000 nodes storing over 10 PB of data, Netflix (2,500 nodes, 420 TB, over 1 trillion requests per day), Chinese search engine Easou (270 nodes, 300 TB, over 800 million requests per day), and eBay (over 100 nodes, 250 TB).

NoSQL

- NoSQL数据库（键值、列式存储、文档、图片）
- 列式存储（HBase）
- 键值存储（Redis、HBase、Cassandra）

RDBMS vs Cassandra

功能	RDBMS	Cassandra
单点故障		
多数据中心		
线性扩展		
数据模型		

ACID vs CAP

ACID（针对关系型数据库事务）

Atomic - All or none（全部成功 或 全部失败）
Consistency - Only valid data is written（仅有效数据能落库）
Isolation - One operation at a time（事务顺序执行）
Durability - Once committed, it stays that way（提交即永久）

ACID vs CAP 续

CAP - 3选2（针对分布式数据库系统）

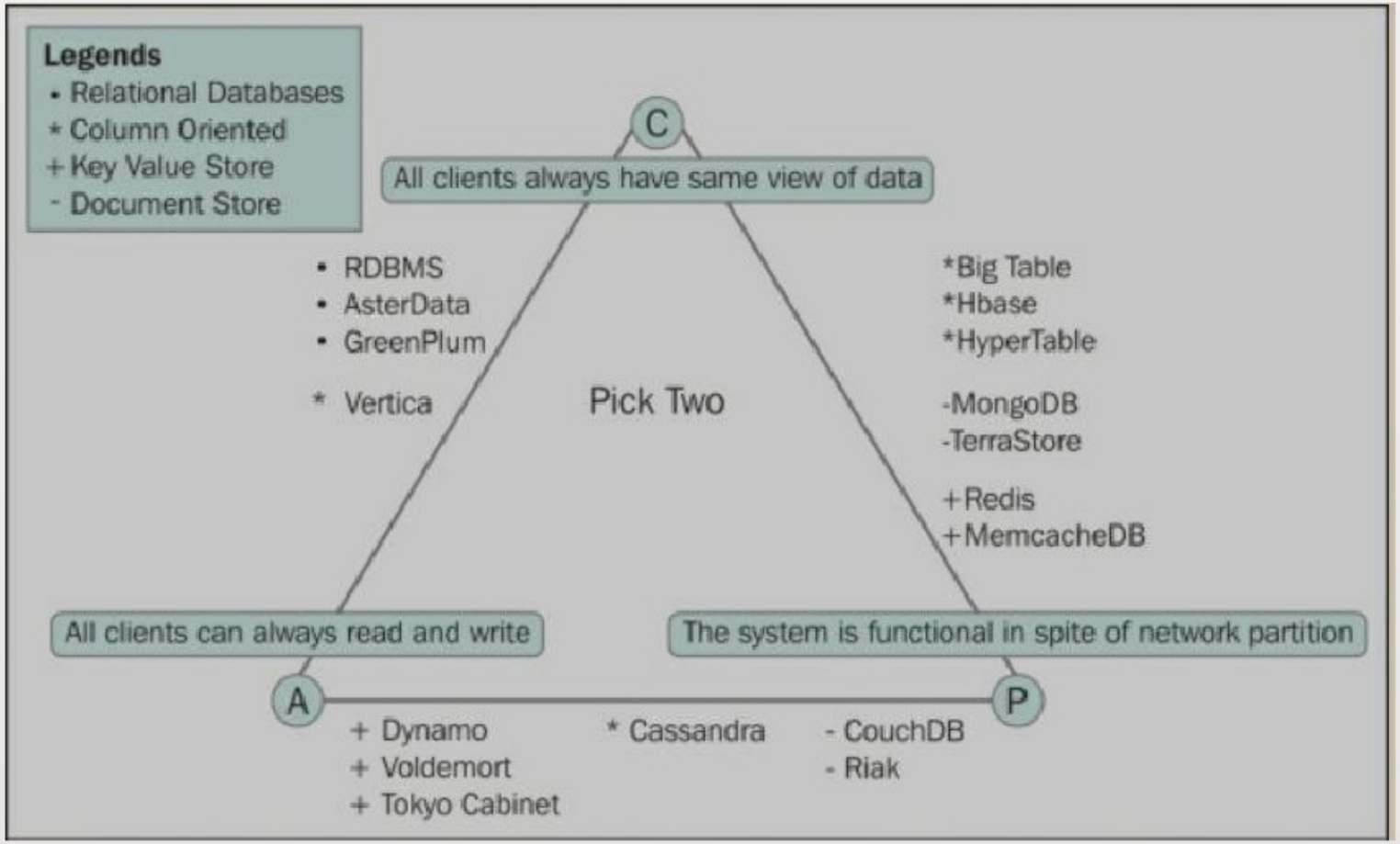
此C非彼C

Consistency - 每次读都能读到最新的数据；所有用户读到的数据是一致的；数据在所有节点上是一致的
Availability - 集群永远可用；能够响应读写请求；读总能找到一个可用的副本，不一定是最新的
Partition tolerance - 能够应对局部网络问题，比如某些节点之间不可达

存在网络分区时，C和A只能二选一

一致性越强，延迟、性能越差

CAP


Legends
• Relational Databases
* Column Oriented
+ Key Value Store
- Document Store
C
All clients always have same view of data
• RDBMS
• AsterData
• GreenPlum
* Vertica
*Big Table
*Hbase
*HyperTable
-MongoDB
-TerraStore
+Redis
+MemcacheDB
Pick Two
All clients can always read and write
The system is functional in spite of network partition
A
P
+ Dynamo
+ Voldemort
+ Tokyo Cabinet
* Cassandra
- CouchDB
- Riak

BASE

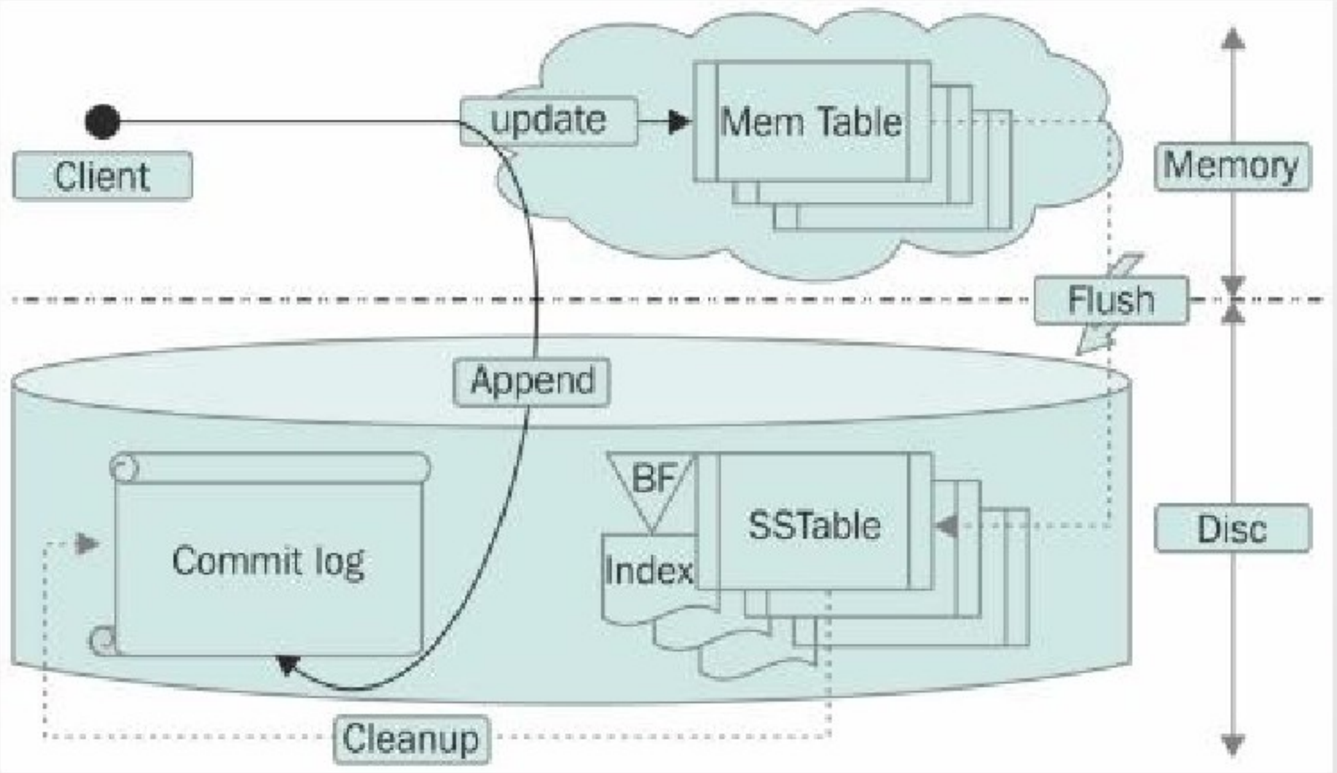
Basically **A**vailable, **S**oft-state, **E**ventual consistency

Cassandra拥抱A&P，可灵活配置实现C

如何达成一致性：
Read repair：读的时候发现不一致，执行修复
Write repair：写的时候发现不一致，执行修复
Asynchronous repair：非读非写时，定期进行修复

基本原理 - 数据存储

- commit log
  - 文件末尾追加；能重放
- memtable
  - 驻留内存
- sstable(sorted string table)
  - 创建后不可更改
  - 小文件合并成大文件



memtable写到磁盘sstable

```
2018-11-20 09:44:41,771 Memtable.java:352 - Writing Memtable-batchlog@1575400604(15.819MiB serialized bytes, 6245 ops, 2%/0% of on/off-heap limit)
2018-11-20 09:45:41,781 Memtable.java:352 - Writing Memtable-batchlog@2129741890(16.181MiB serialized bytes, 6365 ops, 2%/0% of on/off-heap limit)
2018-11-20 09:46:23,755 Memtable.java:352 - Writing Memtable-eventdata@73126797(47.648MiB serialized bytes, 1606545 ops, 31%/0% of on/off-heap limit)
2018-11-20 09:46:41,792 Memtable.java:352 - Writing Memtable-batchlog@1780896487(17.285MiB serialized bytes, 6735 ops, 2%/0% of on/off-heap limit)
2018-11-20 09:47:41,805 Memtable.java:352 - Writing Memtable-batchlog@919335271(15.027MiB serialized bytes, 5955 ops, 2%/0% of on/off-heap limit)
2018-11-20 09:48:41,816 Memtable.java:352 - Writing Memtable-batchlog@1628567682(16.980MiB serialized bytes, 6515 ops, 2%/0% of on/off-heap limit)
2018-11-20 09:49:41,828 Memtable.java:352 - Writing Memtable-batchlog@1129719264(16.288MiB serialized bytes, 6400 ops, 2%/0% of on/off-heap limit)
2018-11-20 09:50:41,839 Memtable.java:352 - Writing Memtable-batchlog@796175075(16.221MiB serialized bytes, 6390 ops, 2%/0% of on/off-heap limit)
2018-11-20 09:51:20,847 Memtable.java:352 - Writing Memtable-eventdata@1606429674(47.768MiB serialized bytes, 1609695 ops, 31%/0% of on/off-heap limit)
2018-11-20 09:51:26,214 Memtable.java:352 - Writing Memtable-compactions_in_progress@1457229695(0.146KiB serialized bytes, 9 ops, 0%/0% of on/off-heap l
```

基本原理 - 副本

副本因子（RF）- 数据在集群有几个副本

写一致性 - 成功写入几个副本算作成功

读一致性 - 成功读取几个副本算作成功

集群节点数： 10
RF： 3
写一致性(Write CL)： LOCAL_QUORUM
读一致性(Read CL)： LOCAL_QUORUM

强一致性：Write CL + Read CL > RF
2 + 2 > 3

```
CREATE KEYSPACE test WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '2'}
```

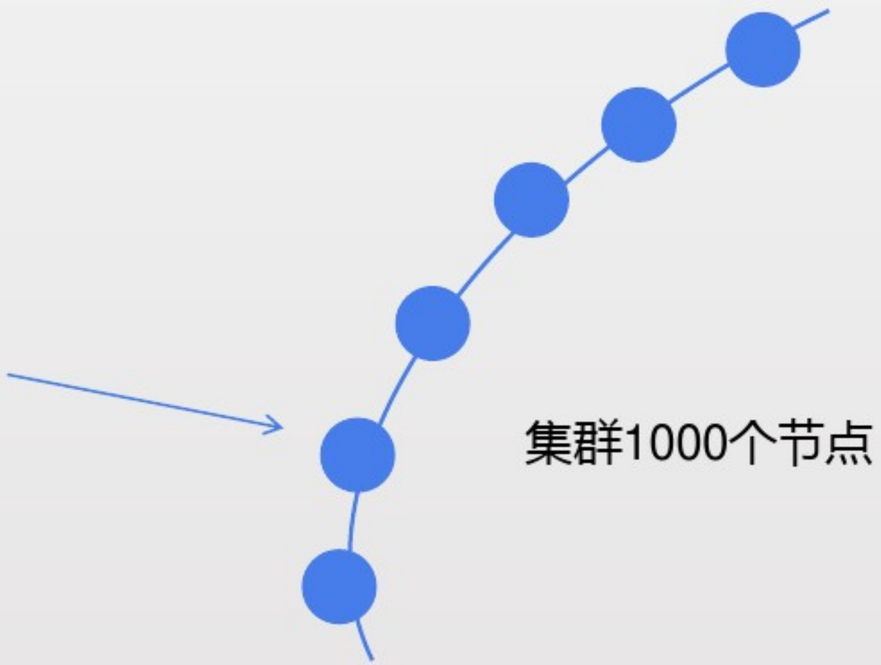
基本原理 - 分区键

SELECT lon, lat, speed, heading

FROM gps_data

WHERE VIN = 868120133697190

VIN = 868120133697190



集群1000个节点

根据分区键计算token

根据token知道数据落在哪个节点

分区键的用途：知道数据在哪个节点上

底层存储


一个设备一天所有的点
device_id + date
时间戳：字段Map
...
时间戳：字段Map
时间戳：1542607426000
lat: 30.2345
lon: 120.120023
speed: 120
heading: 35
altitude: 300

基本原理 - 写机制

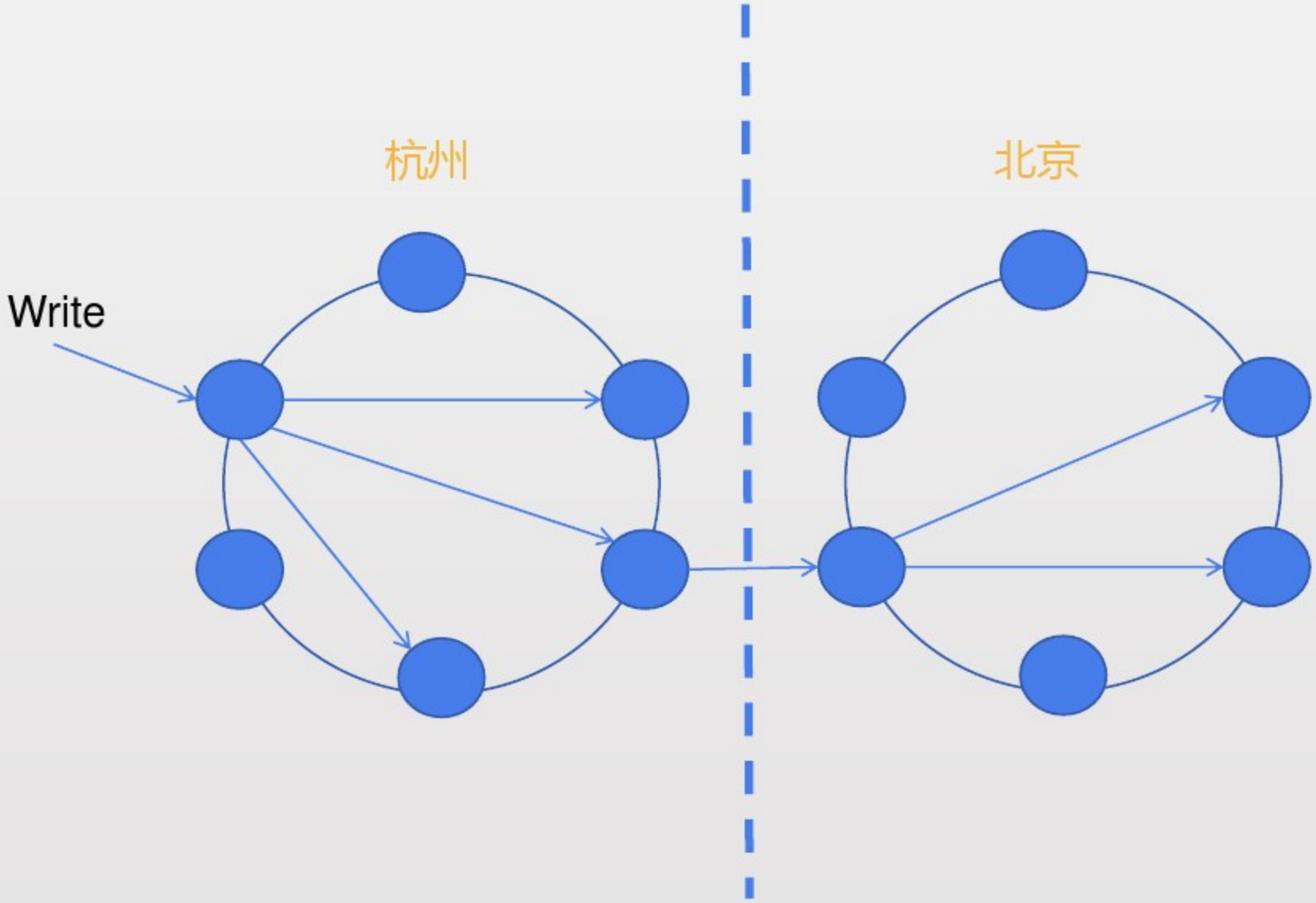
- 接收到写请求的节点成为协调节点（任何节点都可接收请求）
- 根据写一致性要求，将数据转发到不同的节点
- 本地先写commit log，再更新memtable
- 收到其他几点的响应后，将结果返回给客户端

基本原理 - 读机制

- 接收到读请求的节点成为协调节点（任何节点都可接收请求）
- 根据读一致性要求，从选中的节点获取副本数据
- 从其他节点收到副本后，经过计算，将结果返回给客户端

部署方式 - 多数据中心

- 两个数据中心，在线业务和离线分析
- 两地三中心，用于灾备
- 数据中心之间准实时同步



谢谢！